TRACKING CERTIFICATE MISISSUANCE IN THE WILD

BY

ZHENGPING WANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Associate Professor Michael Donald Bailey

ABSTRACT

Certificate Authorities (CAs) are responsible for delegating trust in the TLS Public Key Infrastructure (PKI). Unfortunately, there is a long history of CAs abusing this responsibility, either due to negligence or in some cases, falling victim to attacks. As a result, the PKI community has established standards that define the correctness of certificates and how a well managed CA should operate. In this work, we evaluate a systematic approach to identifying whether certificates issued by CAs are compliant with community standards. To this end, we present ZLint, a system that determines whether a certificate is not conformant to standards, i.e., misissued. We find that while misissuance has decreased over time, there is still a long tail of non-conformant CAs in the ecosystem. Further, our results show that certificate misissuance serves as a reasonable indicator for mismanagement and untrustworthiness, suggesting that CAs that misissue more frequently pose a greater threat to security of the PKI. Community efforts thus far to curb these threats have been moderately successful, but the lack of a systematic approach to identifying these problems lets some classes of problems slip through the cracks. We argue that an automated and systematic approach to measuring misissuance in the ecosystem is a necessary first step in solving the problems that lie ahead.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In the last year alone, there have been a number of high-profile cases where Certificate Authorities (CAs) were caught misbehaving. For example, in September 2016, WoSign, a popular Chinese CA, and their subsidiary, StartCom, were found backdating SHA-1 certificates to avoid updating their systems to deprecate SHA-1. This was an explicit violation of community standards. As a result, both Mozilla and Google have begun the process to distrust WoSign and StartCom certificates in both Firefox and Chrome [1, 2].

WoSign is a recent example of a historically reoccurring problem [3–5]. In light of these failures, the PKI community responded by leading a number of efforts designed to improve the transparency, security, and reliability of the ecosystem. One effort, the CA/Browser Forum Baseline Requirements (BRs), dictates a set of rules that specify the correctness of certificates and how a well-managed CA should operate [6]. Deviations from these standards are not necessarily direct vulnerabilities; however, the community views conformance as a necessary precondition for establishing a secure and reliable PKI [7]. In addition, certain classes of violations do directly lead to vulnerabilities, further increasing the importance of imposing such standards on the ecosystem. Another effort, Certificate Transparency (CT), aims to reduce the opacity of the ecosystem by requiring that trusted CAs submit every certificate they issue to a public ledger. By April 2018, certificates that are not in at least two CT logs will be untrusted by Google Chrome [8].

Despite the successful efforts to establish standards and enable transparency in the CA ecosystem, the community failed to build tools that systematically analyze the data collected. As a result, there is a large effort to utilize the Mozilla Dev Security Policy (MDSP) [9] forum as a means for discussion around misbehaving CAs. Unfortunately, reporting and investigation is done in an ad-hoc fashion, and misissued certificates, which are certificates that do not conform to the BRs, are often found by chance, indicating a need for a systematic approach to identify misbehavior.

We present and deploy ZLint, a system that determines if a certificate is compliant with community standards. ZLint now runs as a part of Censys [10] and `crt.sh` [11], two popular tools used by the PKI community to track certificates. We leverage a corpus of 240M certificates collected through Internet-wide scans and CT logs, active probing of CA infrastructure, and web crawls of MDSP forum posts, to inform a holistic analysis of the ecosystem today.

We track certificate misissuance longitudinally and find that in 2017, only 0.02% of trusted certificates are misissued. This has fallen from 8.4% since the BRs took effect in 2013, and indicates that the ecosystem is certainly improving over time. Still, we find 140K currently trusted certificates are misissued today, and 60% of trusted intermediates on the Internet misissue at least 10% of their certificates. MDSP discussions catch some of these problems, however, we observe discussion is skewed toward larger players and mostly focused on misissuance, rather than all aspects of CA operation.

We next turn to studying misissuance today. We use two metrics—misissuance by raw number of certificates and misissuance by fraction of total certificates issued, to track problematic organizations in the ecosystem. We use recent incidents of CAs losing browser trust (Symantec, WoSign, StartCom, PROCERT) [2, 12, 13] as a guide, and identify two distinct equivalence classes of organizations: ones that have 10,000 or more misissued currently valid certificates, and ones that misissue 90% or more of their total certificates. We find one other instance of the first class (GoDaddy) and 38 other instances of the second, pointing to organizations that should be subject to further scrutiny.

Finally, we investigate how certificate misissuance occurs. We find that most large organizations are in charge of several intermediate CAs, with an average of 20 intermediates below an organization. In 80% of the cases where such an organization misissues a certificate, we can trace the majority of misissuance to a single intermediate within that organization, indicating to us multiple, disparate systems that issue certificates. We argue this adds complexity to the issuance process, and further, makes oversight over an organization complicated, which can lead to further problems.

We conclude with a discussion of the effectiveness of our measurements and propose areas for future work. Our goal is that the results presented in this work serve as a call to action to utilize a systematic approach to analyzing the CA ecosystem moving forward, and that our work will aid in improving the PKI at large.

CHAPTER 2

BACKGROUND

In this chapter, we provide a brief background on TLS, Certificates, and the terminology used throughout the thesis.

2.1 SSL/TLS

Transport Layer Security (TLS) is a session-layer protocol designed to enable end-to-end security for many application layer protocols on the Internet. Using TLS, two hosts create an encrypted channel based on a shared secret, agreed upon by both parties during a handshake protocol run at the beginning of each connection. TLS supports multiple protocols for encryption, and hosts negotiate during the handshake process to select the optimal encryption scheme supported by both the client and server. Although the most prominent use of TLS is as the security layer for HTTPS traffic, it provides the basis for securing other popular protocols, such as SMTP (SMTPS) and FTP (FTPS). To protect against Man-in-the-Middle or impersonation-based attacks, TLS offers the ability to authenticate a remote host using certificates. This technique is most commonly used by clients to verify the identities of servers but can work in either direction.

2.2 PKI and Certificates

In order to manage trust relationships at scale, TLS requires that a Public Key Infrastructure (PKI) be in place and kept up to date. Part of the TLS PKI includes Certificate Authorities (CAs), which are responsible for verifying the identities of entities on the Internet and issuing certificates accordingly. When a client makes a TLS request to a server, the server sends its certificate, which is signed in advance by a CA. If the client trusts the CA, it will trust the server's asserted identity. In practice, this verification process generally relies on a chain of trust, where a root CA A trusts a subordinate CA B, which in turn trusts another subordinate

CA C, which can then sign end-entity, or leaf certificates. If a browser trusts the root CA A, it can use this chain to establish trust of a certificate signed by CA C.

Certificates exist primarily to verify a link between a public key and an named identity. However, in practice, *X.509* certificates are more complex. In addition to public key information, *X.509* certificates contain other data, such as the domain(s) for which a certificate is issued, and issuing CA information. *X.509* also allows for certificate extensions, which are additions to certificates that contain auxiliary data. For example, the `extKeyUsage` extension contains information about what a certificate is used for, like server authentication or email. RFC 5280 [14] outlines these fields and their restrictions in more detail.

2.3 Adhering to Standards

Centralized standards for certificates and CA management guarantee a base level of stability, interoperability, and security. Two major documents specify the community standards with regards to certificates and CA operation: the CA/Browser Baseline Requirement documents (BRs) [6] and the *X.509* RFC [14]. In addition, the BRs contain requirements on CA management—for example, they define how a CA should store and retrieve their keying information and how a CA must operate their revocation systems. Adherence to these standards is not necessarily related to security, but rather related to compliance. However, it is the belief of the PKI community that conforming to these well-defined rules is a necessary precondition for a reliable and secure ecosystem.

2.4 Issuers

We analyze misissuance by aggregating CAs into several categories. For consistency throughout this work, we use the following definitions for each term:

1. *Intermediate*: This is the specific signing entity that issues trusted certificates on the Internet. We identify these by unique issuer common names in trusted certificates.

2. *Organization*: This is the organization responsible for each intermediate in our dataset. We identify these by the unique issuer organization field in trusted certificates.

3. *Business Owner*: This is the business entity responsible for each intermediate in our dataset. A public record of business owners is operated by Mozilla, as a part of the Common CA Database (CCADB) [15].

4. *Root CA*: This is a certificate that exists in the NSS root store.

CHAPTER 3

ZLINT

3.1 Overview

In order to determine whether *X.509* certificates in our dataset comply with the requirements set forth by RFC 5280 and the BRs, we built a new system, called ZLint that checks conformance to the RFC 2119 clauses (SHOULD, MUST, etc.) that appear in each document. We manually parsed the clauses that appear in each document and categorized them based on their function. As an example, the top categories of BR clauses can be seen later in Table 4.2. At the time of this writing, ZLint covers 95% of the certificate related clauses in the BRs and 90% of the clauses in the RFC. ZLint operates on the published version of RFC 5280, and the BRs version 1.4.8. ZLint is implemented in Go as it is part of the Zmap [16] tool chain, and thus consistent with its choice of programming language.

ZLint contains 12,000 lines of code, and can process our corpus of 240M certificates in six hours on five servers, each with 32 cores. ZLint is written in Go, and available as both a standalone system and a pluggable Go library. We worked with the Censys team to integrate ZLint into Censys, and it currently runs on each certificate that Censys finds in the wild. In addition, ZLint was recently adopted by `crt.sh`, a popular tool run by COMODO used to view certificates that appear in CT [11].

3.2 Architecture

ZLint has two levels of failed tests, *Errors* and *Warnings*. Each level is determined by the severity of the clause—MUST clauses map to *Errors*, while lesser severe clauses, like SHOULD clauses, map to *Warnings*. Clauses that ZLint checks come in pairs of a lint and a test. Each clause is implemented in its own lint file, with a descriptive lint name as the file name, e.g. "lint_cert_extensions_version_not_3.go" as shown in Listing

```go
type InvalidCertificateVersion struct{}

func (l *InvalidCertificateVersion) Initialize() error {
    return nil
}

func (l *InvalidCertificateVersion) CheckApplies(cert *x509.Certificate) bool {
    return true
}

func (l *InvalidCertificateVersion) Execute(cert *x509.Certificate) *LintResult {
    if cert.Version != 3 {
        return &LintResult{Status: Error}
    }
    return &LintResult{Status: Pass}
}

func init() {
    RegisterLint(&Lint{
        Name:          "e_invalid_certificate_version",
        Description:   "Certificates MUST be of type X.509 v3",
        Citation:      "BRs: 7.1.1",
        Source:        CABFBaselineRequirements,
        EffectiveDate: util.CABV130Date,
        Lint:          &InvalidCertificateVersion{},
    })
}
```

Listing 3.1: An example lint written in Go.

```go
func TestExtsV2(t *testing.T) {
    inputPath := "../testlint/testCerts/certVersion2WithExtension.pem"
    expected := Error
    out := Lints["e_cert_extensions_version_not_3"].Execute(ReadCertificate(inputPath))
    if out.Status != expected {
        t.Errorf("%s: expected %s, got %s", inputPath, expected, out.Status)
    }
}

func TestExtsV3(t *testing.T) {
    inputPath := "../testlint/testCerts/caBasicConstCrit.pem"
    expected := Pass
    out := Lints["e_cert_extensions_version_not_3"].Execute(ReadCertificate(inputPath))
    if out.Status != expected {
        t.Errorf("%s: expected %s, got %s", inputPath, expected, out.Status)
    }
}

func TestNoExtsV2(t *testing.T) {
    inputPath := "../testlint/testCerts/certVersion2NoExtensions.pem"
    expected := Pass
    out := Lints["e_cert_extensions_version_not_3"].Execute(ReadCertificate(inputPath))
    if out.Status != expected {
        t.Errorf("%s: expected %s, got %s", inputPath, expected, out.Status)
    }
}
```

Listing 3.2: An example test written in Go.

3.1. For each lint file there is also a test file with the same name with an extra "_test" at the end to ensure the correctness of lint implementation, e.g. "lint_cert_extensions_version_not_3_test.go". This is the standard naming convention using Golang's testing framework.

Golang is not designed as an object oriented programming language. Among many other reasons, the developers believe features like multi-inheritance present in C++ is a plague. However, for the purpose of the ZLint project, this object oriented feature is desired. All lints share common functionalities such as initialization, validity check and etc. Thus we decided to mimic the object oriented feature using interfaces, where each specific lint implements a generic base lint interface.

3.2.1 Lints

To understand how ZLint works under the hood, we will explain what each part of the example lint code in Listing 3.1 does. The *struct* following the lint name is used to store data structures and variables that are specific to that lint. For example, an RSA related lint checks the public exponent of a public key against a scientifically big constant upper bound value. It is reasonable to store this constant as a static member of the lint, since each lint will be run against millions of certificates, and the computation of the upper bound can be computed only once. We introduce the following functions as a part of each lint.

Initlialize() is used to initialize aforementioned lint-specific data structures.

CheckApplies() determines if it is applicable to run a lint against the input certificate. For example, if the certificate is using DSA as the signing algorithm, it does not make sense to run it against RSA lints.

Execute() invokes the body of the lint that carries out the actual linting.

RegisterLint() registers the lint into a global table, so that some time later a certificate can be passed to all the lint functions in order. *Name* starts with either an "e" or a "w", representing error and warning respectively, and the rest of the name acts as an identifier to what the lint is for. *Description* is a short

excerpt from the documents describing a SHOULD or MUST clause. *Citation* specifies what section of the document description comes from. *Source* specifies what document the description comes from.

EffectiveDate represents the date from which a lint is required by the standards. If a certificate is valid before this date, the lint will not be checked against this certificate.

Consider the following example clause from the BRs section 7.1.1:

> "Certificates MUST be of type X.509 v3."

For this clause, we implement a lint that checks whether the version number presented in the certificate is equal to version 3. If not, we return an *Error*. An example of the code required for this lint is shown in Listing 3.1. If a certificate fails a test that returns an *Error*, it is deemed misissued, which means the CA was not allowed to issue that certificate for that entity. A *Warning* is not as severe, but indicates that a CA does not follow a recommendation set forth by the requirements for that certificate.

3.2.2 Tests

ZLint tests use Golang's built-in testing framework. Test files contain multiple test functions, each of which uses a different input certificate. As shown in Listing 3.2, *inputPath* indicates where the input certificate is located.

expected is the expected outcome of a lint running against a certificate; it could either be PASS, WARN or ERROR.

out is the actual outcome we get from running the lint on the certificate. When *expected* and *out* do not match with each other, the testing framework will throw an error to stdout.

Lints is the global table where a lint name is mapped to the lint function.

For the test certificates, we first create a template certificate using *CreateCertificate* function in the x509 package. This allows us to build certificates with any arbitrary attributes and extensions with a few

exceptions. For example, the Golang *Time* package refuses to create GeneralizedTime objects without seconds, or with a fraction of seconds. One of the lints is designed to catch certificates with these non-compliances against RFC. In this case, we inspect the ASN.1 encoded certificate content and change those values accordingly by hand. The template certificate is built in a way such that it is expected to pass specific lint. If it actually fails the test, we then know that there are flaws in our implementation. Otherwise, we proceed to mutate the passing certificate by a bit, in particular, changing the fields or extensions that the lint is concerned with. The resulting certificate should violate the standard documents in one way or another and fail corresponding tests. As shown in Listing 3.2, three tests are created for the certificate version lint. According to RFC 5280, "when extensions are used, as expected in this profile, version MUST be 3 (value is 2)." The first test uses a certificate that has extensions and is of version 2, which violates the clause. The second test uses a certificate that has extensions and is of version 3, which complies with the clause. The third test uses a certificate that has no extensions and is of version 3, which also complies with the clause.

3.2.3 ZLint Workflow

Input file to ZLint can be either of ".pem" or ".csv" extensions containing public certificate files. If the input is one single certificate, it should be a ".pem" file, otherwise it should be a ".csv" file with certificates separated by newlines. The certificate content is ASN.1 DER encoded and stored in base64 format. The *main* function takes in raw certificates and parses them into the Certificate struct defined in the Golang x509 package. Afterward, lints get registered into a global table and applied to each certificate over all parsed certificates. The output consists of JSON fields with lint names as keys and pass, warn, or error as values. Finally JSON outputs are aggregated and written to output files.

3.3 Runbook

ZLint is open-source and available at https://github.com/zmap/zlint.

Zlint can be installed as a standalone command-line tool using the "go get" command

$go get github.com/zmap/zlint/cmd/zlint

To run ZLint against a certificate, one could do

$zlint mycert.pem

ZLint can be also used as a library. One could simply import our path.

import “github.com/zmap/zlint”

Adding new lints is as simple as copying and following the structures of an existing lint, and changing the names to match what the new lint does, and modify the body of the lint function as expected.

CHAPTER 4

METHODOLOGY

We utilized a corpus of 240M certificates collected from Internet-wide scans and Certificate Transparency logs, active measurements, and web crawls of MDSP to aid our analysis of the CA ecosystem. Table 4.1 presents a high-level summary of the data presented in this thesis.

4.1 Gathering Certificates

Part of our analysis focuses on the X.509 certificates that chain to a root in the NSS root store, which is used by Mozilla Firefox. To capture a comprehensive representation of such certificates, we relied on Censys [10], which has collected certificates from daily IPv4 HTTPS scans since 2013. Censys also includes certificates found in Certificate Transparency (CT) logs, making it the largest collection of certificates available to date [17]. In total, the dataset includes roughly 240M certificates. Many of these certificates, however, are either expired or untrusted by default (imagine a self-signed cert presented by a router found in a scan). After removing them, we are left with 61M certificates that are currently trusted by the NSS root store, and 170M certificates that were trusted at a prior point in time. There are 1320 unique intermediate issuers and 618 unique issuing organizations in our dataset.

Table 4.1: **Data Sources**—We utilized a variety of data perspectives to analyze the behavior of CAs.

Role	Data Source	Collection Period	Data Volume
Certificate Misissuance	Scanning + CT	2013–2017	240M certificates, 1320 intermediates, 618 organizations
	ZLint		12K LOC, 224 tests
CA Management	OCSP Responses	09/01/2017–09/20/2017	454 probes, 1419 responders
	CRL Responses	09/01/2017–09/20/2017	454 probes, 3361 CRL URLs
Public Profile of CA	MDSP Forum	01/05/2015–09/19/2017	6069 posts, 630 CA mentions

Table 4.2: **Categorization of the Baseline Requirements**—We show the various categories of BR requirements. The largest number of requirements appear on the correctness of certificate extensions fields, which many applications rely on.

Category	BR Sections	# Clauses
Extensions	7.1	47
Certificate Attributes	6,7,8	36
Identity Validation	1,3,8	20
CA Management	2,5	20
Revocation	4	11

4.2 MDSP

The Mozilla Dev Security Policy (MDSP) forum is an online forum where members of the PKI community discuss topics related to the HTTPS PKI, as well as report instances of misissuance and other CA misbehavior [9]. As such, MDSP acts as a "public profile" of a CA. We used the number of times a CA was mentioned doing something unexpected as a proxy for "CA Notoriety" in the community. We crawled the MDSP forum from the beginning of 2015 and searched for links to certificates that were misissued. We then augmented this list by manually inspecting each post for other kinds of mentions that did not include certificate links. In total, we parsed 6,069 posts and comments, and aggregated a total of 630 CA mentions.

4.3 Organizational Management

As part of operating a CA, organizations must often run systems that supplement their issuance of certificates in the PKI. In order to glean insight into CA management, we measure three of these systems through active measurements in the wild.

4.3.1 OCSP Responder Health

An Online Certificate Status Protocol (OCSP) responder is a system that CAs are required to run in order to support online revocation checking for certificates they have issued. We actively probe the trusted OCSP responders on the Internet to measure their health and their correctness. We first collected a list of OCSP responders for each CA, which are embedded in certificates. This resulted in 1,419 responders. We then ran three experiments designed to check whether the OCSP responders behaved properly.

1. OCSP responders are required to be live on a 24x7 basis. We constructed a valid OCSP request for each responder in our dataset and sent this request every hour between September 1st and September 20th, 2017. We observed when OCSP requests timed out and how long they took.

2. Per the BRs 4.9.10, an OCSP responder should update subscriber certificate information at least every four days. For each correctly formed OCSP request we made, we checked whether the validity periods conformed to the standards.

3. Per BRs 4.9.10, an OCSP responder MUST NOT return a "GOOD" status code for certificates it has not previously issued. To test this, we constructed an OCSP request for each CA with the serial number `deadbeefdeadbeef`, which we found no CA had issued in our dataset, and sent it to each responder in our dataset.

4.3.2 CRL Maintenance

Although OCSP is a preferred revocation protocol, Certificate Revocation Lists (CRLs) are still supported by CAs who have customers that rely on them for revocation information. If a CA supports a CRL, it must be offered as a 24x7 service and respond within a reasonable time to requests. We collected a list of CRL URLs for each CA, which totaled in 3,361 CRL URLs. We made a valid request to each CRL URL every hour between September 1st and September 20th and observed how long each request took as well as how frequently CRL requests timed out.

CHAPTER 5

AN EVOLVING ECOSYSTEM

The certificate ecosystem has evolved since its last study in 2013 [18]. To contextualize our analysis of certificate misissuance, we first provide an update on the ecosystem and the efforts that have risen since its 2013 study. We then present a longitudinal view of certificate misissuance and use it as a starting point for further analysis.

5.1 Certificate Authorities

Table 5.1 shows the distribution of top organizations that issue trusted certificates in our dataset. We find 613 such organizations in our dataset, which is comparable to the number that were present in the ecosystem in 2013 (683) [18]. Similarly to 2013, we find that the top five organizations make up for approximately 87% of the total certificates issued today, however, the distribution of these issuers has shifted. Most notably, Let's Encrypt was launched in 2016 as a free, usable CA, and has risen to prominence in the ecosystem. Let's Encrypt issues 36M currently trusted certificates, which accounts for 61% of trusted certificates issued (Table 5.1). This is an order of magnitude larger than the next-largest organization, COMODO, which issues 6.7M certificates, or 11.2% of currently trusted certificates on the Internet.

Table 5.1: **Top Trusted Issuer Organizations**—We show the top certificate issuers in our dataset. *Let's Encrypt* vastly overshadows its competitors, with an order of magnitude more certificates issued compared to the next-largest organization, *COMODO*.

Issuer	Current Valid Certificates
Let's Encrypt Authority X3	37M (61.1%)
COMODO CA Limited	6.7M (11.2%)
cPanel	4.7M (7.8%)
Symantec Corporation	2.8M (4.6%)
GeoTrust Inc.	1.9M (3.2%)
GoDaddy.com	1.6M (2.7%)
GlobalSign nv-sa	1.2M (1.9%)

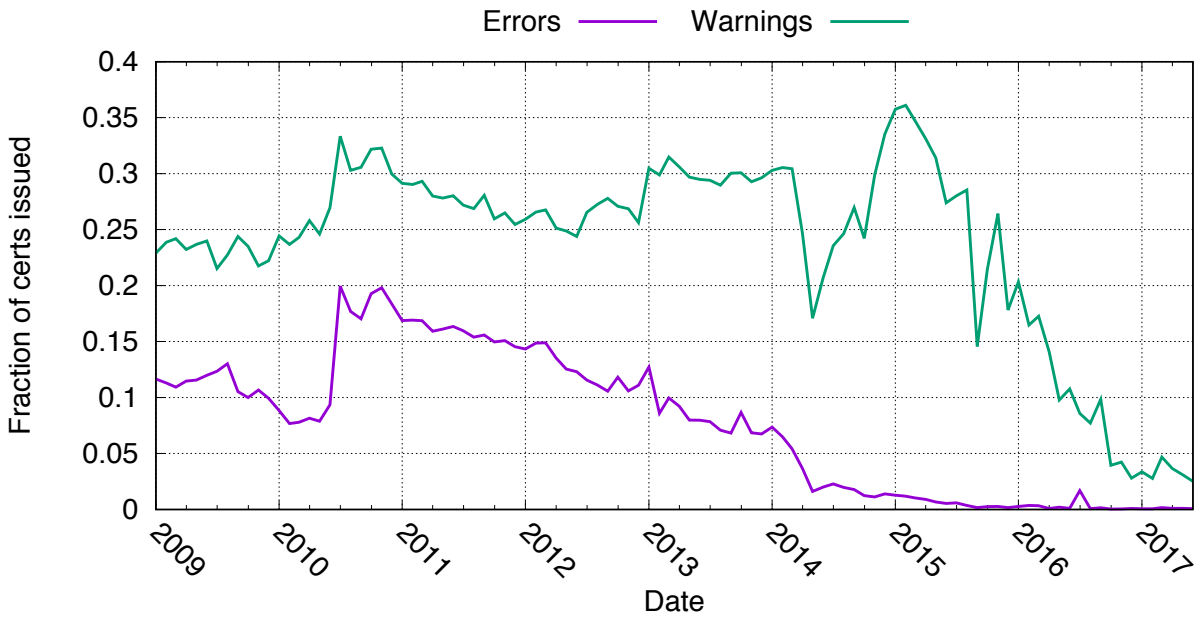


Figure 5.1: **Certificates Issued with Errors and Warnings**—The fraction of certificates issued with errors and warnings by trusted organizations has decreased over time, and dropped 80% since 2014. In 2017, only 0.02% of certificates are misissued with errors, and 3% are issued with warnings.

Table 5.2: **Certificates Issued with Errors and Warnings**—The fraction of certificates misissued with errors by trusted intermediates has decreased over time and dropped rapidly after 2014. In 2017, only 23K certificates are misissued, out of a total 102M so far.

Year	# Error Certs	% Error Certs	# Total Certs
2012	287,454	12.4%	2,321,127
2013	240,943	8.4%	2,875,293
2014	101,631	2.9%	3,557,671
2015	35,419	0.5%	7,037,597
2016	24,008	0.04%	50,420,144
2017	23,207	0.02%	102,187,984

5.2 New Efforts

Since 2013, there have been a number of concerted efforts toward improving the security and reliability of the TLS PKI. We discuss four such efforts in detail: the CA/Browser Forum Baseline Requirements (BRs), MDSP, Certificate Transparency (CT), and the rise of Let's Encrypt.

Baseline Requirements: A lack of standardization for the management of a trusted CA gave rise to the CA/Browser Baseline Requirements (BRs), a document which establishes rules for both the management of a CA as well as the correctness of certificates issued in the PKI. We note these requirements only apply to certificates that are used for server authentication, and there are no additional standards besides RFC

5280 that apply to other kinds of certificates. These requirements are ever changing, and new versions of the requirements are ratified approximately every six months. The BRs serve as the primary requirement to maintain browser trust, as compliance to the BRs is seen as a necessary precondition for trustworthiness [7]. We believe the curation of these rules was a critical first step in standardizing the behavior of CAs at large.

MDSP: MDSP [9] is an online forum where members of the PKI community discuss topics related to the TLS ecosystem. This is also the venue that many in the community use to report certificate misissuance—in this sense, MDSP acts as a bridge between CAs and everyday users. Although the forum was created in 2009, it picked up rapidly in 2014 as users reported issues more frequently and CAs published subsequent incident reports. Oftentimes, when a misissuance is discovered, the responsible CA not only revokes the bad certificates, but patches their systems such that the misissuance does not happen again. As an example, on February 26, 2017, members on MDSP found that several certificates issued by GlobalSign were misissued due to poor checking of DNSNames in issued certificates. GlobalSign responded, and revoked all the affected certificates. They also asserted that they deployed better DNSName checks as of February 2016—prior to this date, we found 39 misissued certificates in our dataset, but after the patch, no more such certificates were misissued in that way. Many similar examples of this kind of feedback loop appear on MDSP; they serve as a testament to the positive impact MDSP has on the quality of issued certificates.

Certificate Transparency: Certificate Transparency (CT) is an effort started by Google in 2013 to build and maintain a public record of all trusted certificates on the Internet. Over the years, the project has grown in usage, primarily driven by the fact that Google Chrome will distrust certificates that do not appear in at least two CT logs starting in April of 2018 [8]. As a result of CT, misissuance has become far easier to observe—as soon as a certificate is uploaded to CT, anyone on the Internet can download it and inspect it for non-conformity.

Let's Encrypt: Let's Encrypt launched in 2016 as a CA with a focus on automating the process of certificate issuance and the higher goal of increasing HTTPS adoption. Since their launch, they have issued over 100M free certificates, making them the largest organization by certificate volume. They have 36M currently valid certificates issued, and of these, just 13 are misissued as of July 2017. Better yet, these have

since been revoked as a result of a recent discussion on MDSP [19]. Let's Encrypt has thus been influential on the state of misissuance in the ecosystem as a whole, and acts somewhat as a "gold standard" for certificate quality.

5.3 A History of Misissuance

Given our longitudinal perspective of certificates, we begin with the question: "How has certificate misissuance changed over time?" Figure 5.1 and Table 5.2 show the fraction of certificates that are misissued longitudinally, up through July of 2017. We note that these are certificates that were valid at the time of their issuance but may not be valid today. We find that the fraction of misissued certificates has decreased over time and continues to decrease yearly. In addition, the raw number of misissued certificates is also decreasing over time, despite the rapid growth of certificates issued. From January to July 2017, only 0.02% of certificates have been misissued with errors, and 3% of certificates are issued with warnings. This has fallen from 8.4% since the BRs took effect, indicating a 99.7% decrease in certificate misissuance since 2013. This is an encouraging result, and serves as an indicator that many of the efforts in recent years to improve the state of the ecosystem are indeed working. Unfortunately, there are still 140,000 misissued certificates that are currently trusted, indicating much more work to be done before this problem is solved.

CHAPTER 6

THE ECOSYSTEM TODAY

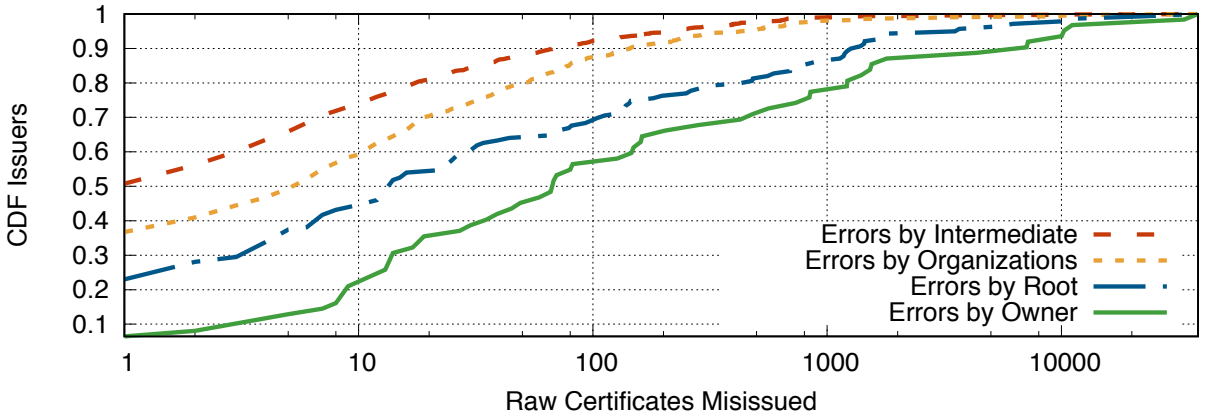
Although the certificate ecosystem as a whole is increasing in quality, there is still much work to be done. Figure 6.1 shows the distribution of misissuance by issuers in our dataset: 40% of intermediates, 50% of organizations, 45% of business owners, and 35% of the root CAs in our dataset misissue at least 10% of their certificates, indicating to us a long tail of non-conforming issuers. These are rarely big players—the average number of certificates issued by intermediates that misissue at least 10% of their certificates is 530, compared to an average of 75K certificates issued by the remaining intermediates.

In order to determine who is responsible for misissuance in 2017, we analyze the issuers that misissue certificates by organization, business owner, and the root CA that intermediates chain to. We investigate misissuance both by the fraction of total certificates misissued the by the total number of certificates misissued to capture a holistic view of non-conformance.

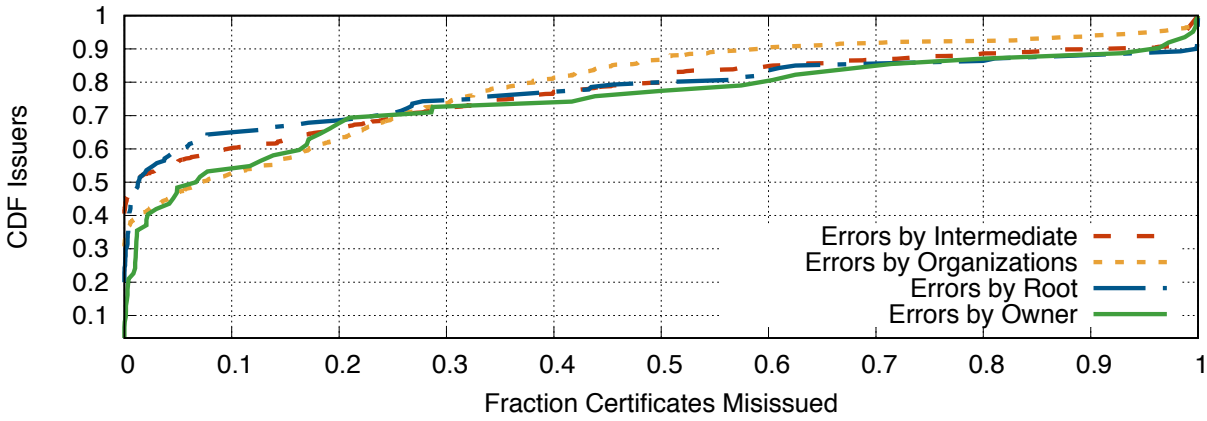
6.1 Misissuance by Organization

Table 6.1 shows the top five issuing organizations by both the fraction and the total number of misissued certificates. We note that organizations that are responsible for the most errors by fraction of certificates issued are often small, and generally issue a total of less than 1000 certificates in our dataset. Certificates issued by these organizations also tend to be issued consistently with the same kinds of errors, indicating a fundamentally broken issuance process that does not conform to standards. This is in contrast with organizations that issue the most number of raw certificates with errors, which tend to be larger, recognizable organizations and issue hundreds of thousands of certificates.

While it may not be surprising that organizations that issue a larger number of certificates tend to misissue more, it is not the case that the raw size of an organization implies it will misissue more certificates. For



(a) **Raw Misissuance by Issuer**



(b) **Fraction Misissuance by Issuer**

Figure 6.1: **CDF by Issuer**—Most intermediates, organizations, business owners, and Root CAs misissue some fraction of certificates on the public Internet.

Table 6.1: **Organizations with Most Errors, by Fraction and Raw Number**—The organizations that are responsible for the most errors by fraction of certificates issued are often small, generally issuing less than 1000 certs. In comparison, the organizations that issue the most raw error certificates are large and often recognizable organizations.

Organization	Error Certs	% Error Certs	# Intermediates
Nestle	968	100%	1
Freistaat Bayern	393	100%	2
Giesecke and Devrient	18	100%	1
Unizeto Sp. z o.o.	18	100%	1
CertiPath LLC	9	100%	1

Organization	Error Certs	% Error Certs	# Intermediates
GoDaddy.com	38,215	2.4%	3
Symantec Corporation	23,053	0.8%	22
StartCom Ltd.	11,617	2.1%	17
WoSign CA Limited	9,849	5.0%	39
VeriSign	9,835	23.1%	10

example, Let's Encrypt has 36M active certificates, but has only misissued 13. The third largest organization by active certificates, cPanel, has issued 4.7M. Of these, only 131 are misissued. What makes these large organizations different from other large players? Oftentimes, large PKI organizations own and operate many CAs for varying purposes. As an example, COMODO, the second largest organization by volume of certs issued, has different intermediates for issuing RSA Extended Validation certs and ECC Extended Validation certs. In total, COMODO has 29 intermediates it utilizes to issue certificates. Table 6.1 also shows the number of distinct intermediates each organization manages. We noticed that the organizations with the largest number of raw errors often managed many intermediates, which led us to the question: "Is there a correlation between the number of intermediates an organization manages and the amount of misissuance?" To calculate this, we first rank order the organizations in our dataset both by the number of intermediates they control and the number of certificates misissued. We then apply the Spearman's rank correlation test, a non-parametric test that measures the statistical dependence between two ranked variables. We find that there is a moderate and statistically-significant correlation between the number of intermediates that an organization manages and the raw certificates misissued, with a correlation coefficient of 0.31 and a p-value of 2.2×10^{-14}. This indicates to us that as an organization grows to support many intermediates, the raw number of certificates misissued also increases. We also find no correlation between the number of intermediates per organization and the fraction of misissuance, which lends credence to the notion of many small organizations that consistently misissue all of their certificates.

6.2 Misissuance by Business Owner

An ongoing effort to enable transparency in the CA ecosystem is the Common CA Database (CCADB) [15], a public record containing information about CAs that chain to a browser trusted root store. As part of its root store policy, Mozilla requires that intermediates that chain to a root in the NSS store be listed and kept up to date in CCADB. A result of this public record is that we can also analyze misissuance by *business owners*, which are often larger and further reaching than organizations. This is because distinct organizations can be operated by the same business owner, either due to a merger between two PKI companies, or an acquisition made by a larger player. For example, Symantec not only operates its own intermediates, but

Table 6.2: **Business Owners with Most Errors, by Fraction and Raw Number**—The business owners that misissue the largest fraction of their certificates are small, often managing just one unique organization. In contrast, the business owners that misissue the largest numbers of raw certificates often have many, heterogeneous sub-organizations.

Business Owner	Error Certs	% Error Certs	# Orgs
PROCERT	39	100%	1
Gov't of Spain (FNMT)	482	99.8%	1
D-TRUST	1,398	99.8%	1
Gov't of Spain (ACCV)	840	99.5%	1
Gov't of Turkey (Kamu SM)	9	98.7%	2
Business Owner	**Error Certs**	**% Error Certs**	**# Orgs**
GoDaddy	38,324	2.1%	2
Symantec / VeriSign	33,436	1.1%	9
WoSign CA Limited	11,132	4.9%	6
Start Commercial (StartCom)	10,259	2.0%	3
Deutsche Telekom	10,028	19.1%	345

also has oversight into intermediates operated by other large organizations, like VeriSign, GeoTrust, and Thawte.

There are 63 business owners responsible for the 600 organizations and 1300 intermediates in our dataset. We note that 80 intermediates in our dataset that do not appear in CCADB, so we could not collect any further information about them. In instances where CCADB pointed to multiple business owners for a given intermediate, we manually visited their certificate policy statement (CPS) to identify the proper business owner. Table 6.2 shows the distribution of business owners with the most errors, again by their fraction of total certificates and raw number misissued. The business owners that misissue the largest fraction of certificates are small, with the owners that misissue more than 90 percent of their certificates issuing an average of 433 currently trusted certificates.

Our finding that the number of intermediates per organization is correlated with the raw amount of misissuance led us to ask a similar question for business owners: "Is there a correlation between a number of organizations a business owner manages and its subsequent misissuance?" In fact, we find that there is a stronger correlation between the number of organizations that a business owner must manage and the raw certificates misissued, with a correlation coefficient of 0.55 and a p-value of 3.2×10^{-6}. Again, we see as a business grows to manage more distinct organizations, the raw number of certificates misissued also increases. The opposite is also true—there is a moderate and significant *negative* correlation between organization size

Table 6.3: **Misissuance by Root CA**—Consistent with our findings about organizations and business owners, we find that roots that misissue the largest fraction of certificates are parents to a smaller number of intermediates when compared to those that misissue the largest raw numbers of certificates.

Root	# Error Certs	% Error Certs	# Intermediates
D-TRUST Root Class 3 CA 2 2009	1134	100%	2
ACCVRAIZ1	836	100%	6
Microsec e-Szigno Root CA 2009	419	100%	18
D-TRUST Root Class 3 CA 2 EV 2009	262	100%	2
TUBITAK UEKAE Kok Sertifika Hizmet Saglaylclc1 - Surum 3	145	100%	3
Root	**# Error Certs**	**% Error Certs**	**# Intermediates**
Go Daddy Class 2 CA	38,151	26.8%	3
VeriSign Universal Root Certification Authority	21,814	1.2%	20
StartCom Certification Authority	10,269	2.0%	52
VeriSign Class 3 Public Primary Certification Authority - G3	9,851	1.4%	30
Certification Authority of WoSign	6,197	5.0%	19

and fraction of certificates misissued, with a correlation coefficient of -0.33 and a p-value of 7×10^{-3}. This reverse correlation paints a similar picture to that of organizations—there are many small business owners that misissue all of their certificates.

6.3 Misissuance by Root

Root CAs are responsible for the delegation of trust in the PKI, and as a result, any intermediate that falls below it in a chain-of-trust reflects on the "trustworthiness" of that root CA. We aim to answer the question: "Which roots misissue more than other roots, and why?" To do this, we aggregate each CA by the root that it chains to in the CCADB documentation. We view the output of a root CA as the sum of its downstream intermediates.

Table 6.3 shows the distribution of roots by both the fraction and the raw number of misissued certificates. It additionally shows the number of intermediates that chain to each root. Consistent with our other findings, we note that the roots that misissue the largest fraction of certificates are small—those that misissue at least 90% of their child certificates only issue an average of 228 certificates. Also consistent with our previous findings for business owners, we find that there is a weak correlation between the number of intermediates below a root CA and the raw number of certificates misissued—a Spearman's rank test gives a correlation of 0.23 with a p-value of 0.006. This indicates that the wider a root spreads its trust, the higher the likelihood

Table 6.4: **Fraction of Misissuance over Time**—We show the three major organizations that have lost browser trust compared to the rest of the ecosystem. In 2016 and 2017, all three organizations were misissuing certificates at a rate 2-8x worse than the remainder of trusted organizations.

Year	Overall	Without LE	Symantec	StartCom	WoSign
2013	8.4%	8.4%	0%	0.3%	0%
2014	2.9%	2.9%	6.7%	0.3%	2.5%
2015	0.5%	0.5%	1.2%	0.3%	0.3%
2016	0.04%	0.2%	2.5%	4.7%	8.6%
2017	0.02%	0.2%	1.9%	0.1%	3.0%

it will act as a root to misissued certificates.

6.4 Losing Browser Trust

In the last year, there have been three incidents where organizations have lost browser trust and been subsequently removed from the NSS root store. In October of 2016, Mozilla released a statement that they would no longer trust new certificates issued by WoSign and StartCom, two popular organizations (WoSign, which owns StartCom, is the 15th largest organizations by currently trusted certs issued), as the community discovered a number of "technical and management failures" serious enough to undermine the trust of the PKI [1]. Then, in March of 2017, Google released a statement on the blink-dev mailing group that as a result of continued issues with the trustworthiness of Symantec, Google Chrome planned to distrust Symantec roots in Chrome 66 and beyond, effectively crippling the fifth largest organization by volume of trusted certificates [12]. Most recently, in September 2017, Mozilla announced a plan to remove PROCERT, a small organization that issues just 39 certificates, from the NSS root store [13].

These incidents motivated us to the following question: "Are there other organizations that currently behave similarly to those that have already been removed?"

Based on Tables 6.1, 6.2, and 6.3, all three of Symantec, StartCom, and WoSign are near the top of raw certificates misissued. Their respective organizations, business owners, and roots misissue more than 10K certificates each, putting them all in the same "equivalence class" of raw misissued certificates. For at least these examples, misissuance acts as an indicator for untrustworthy organizations. There is just one other organization within this equivalence class, GoDaddy, which misissues a total of 38K currently valid certificates in our dataset. Upon further investigation, we find that although these certificates were in fact

misissued—they are missing the required SAN extension—all such certificates were issued in late 2012 and early 2013, and GoDaddy has not misissued certificates in the same way since. All of these certificates will expire by the end of 2017, at which point GoDaddy will fall among the ranks of Let's Encrypt and cPanel as a large issuer with infrequent errors. This is in contrast to Symantec, StartCom, and WoSign, that continue to misissue certificates through 2016 and early 2017.

In addition, Symantec, StartCom, and WoSign also behave worse than the rest of ecosystem in terms of fraction of misissuance. Table 6.4 shows the three organizations in question, with their fraction of misissuance compared to the remainder of trusted organizations in our dataset. We provide two comparisons—one including Let's Encrypt, and one excluding Let's Encrypt, as they vastly outnumber the remainder of organizations and rarely have errors. We find that the three organizations in question do have a quantifiable difference in terms of certificate output—in 2016 and 2017, all three organizations misissue certificates at a rate 2-8x worse than the remainder of the ecosystem. An interesting trend to note is that fraction of misissuance does not necessarily decrease over time in these organizations, for example, although StartCom only misissued 0.3% of their certificates in 2015, they misissue 4.7% the following year. This can occur for several reasons—one of which is that new requirements may come into effect over time. A good example of this is WoSign's certificate output in 2016, which spikes up to 8.6% of all certificates it issues. As of January 2016, a new requirement was put in place disallowing new SHA-1 certificates, which WoSign struggled to adapt to. As a result, they turned to backdating certificates with SHA-1 signatures, which led them to eventually lose browser trust.

A more dire story exists for organizations, business owners, and roots that misissue the largest fraction of their certificates. PROCERT leads business owners with 100% of its certificates misissued, despite only issuing 39 currently trusted certificates. Unfortunately, there are several other organizations and business owners that fall in a similar equivalence class—37 other organizations and 8 other business owners misissue more than 90% of their certificates. With the precedent set by Mozilla regarding PROCERT, we argue that such organizations and businesses should at least be further scrutinized, and if they are not up to par, removed from browser trusted root stores.

CHAPTER 7

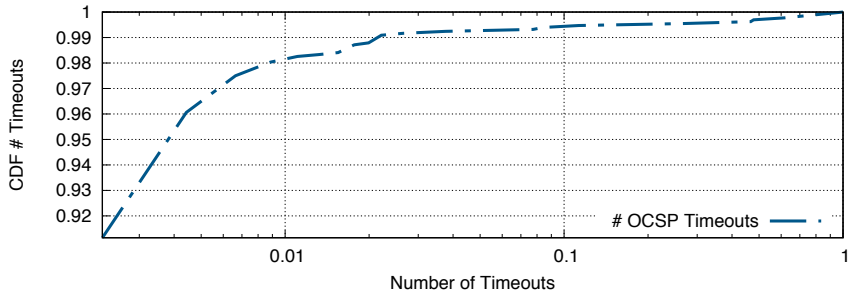
ORGANIZATIONAL MANAGEMENT

In addition to requirements that apply only to certificates, there are many requirements in the BRs that focus instead on how well a PKI organization is managed. Organizations must instrument and operate systems that may not directly impact the issuance process, but are required in a trusted PKI. We check two of these properties, namely, the health and correctness of an intermediate's OCSP responder and the maintenance of their CRL distribution point. We conclude this section with a discussion of consistency and correlation between ZLint, Management, and the public profile of an organization (MDSP).

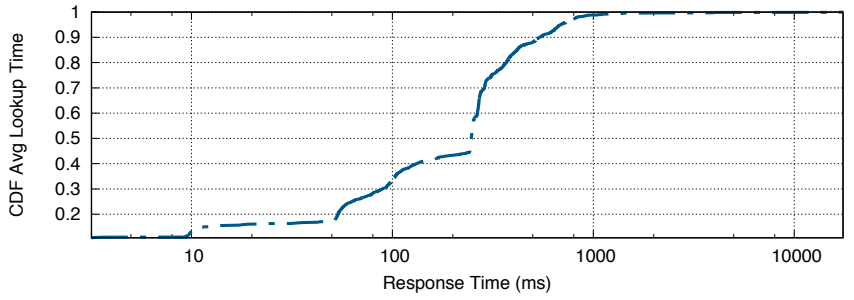
7.1 OCSP Responder Health

Online Certificate Status Protocol, or OCSP, is the preferred mechanism by which entities on the Internet determine the revocation status of a certificate. Organizations are required by the BRs to maintain an OCSP responder that keeps track of the certificates they have issued. An OCSP responder is a critical piece of infrastructure, and the BRs mandate several properties of OCSP responders in order to ensure their correctness and availability. We perform longitudinal measurements against OCSP responders, querying every OCSP responder found in our dataset and comparing their behavior across the various requirements.

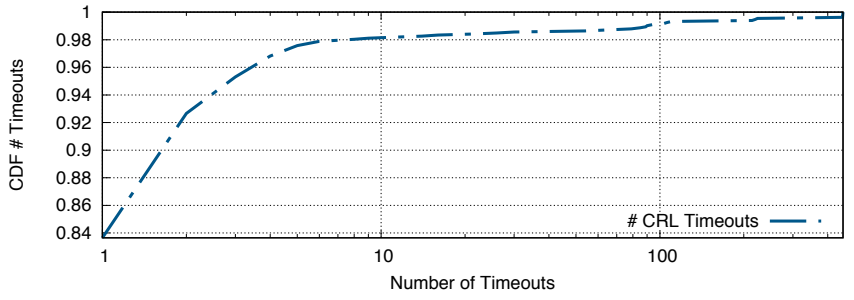
OCSP Availability: The first measurement we conduct is to make a correctly formed OCSP request to every OCSP responder in our dataset. We deem an OCSP responder to "timeout" if it does not respond within 60 seconds. In total, 25% of the total OCSP responders timed out at least once in our study. Among these, there is large variance between responders—Figure 7.1a shows a CDF of the fraction of timeouts. In the worst case, a handful of OCSP responders consistently timeout, indicating fundamentally broken services. The remainder is a long tail of services that timeout infrequently and sporadically, indicating



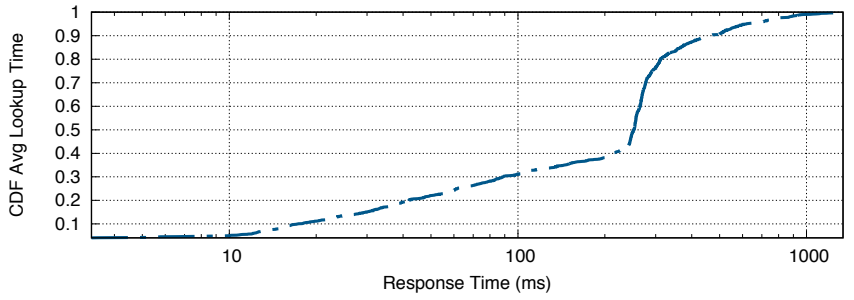
(a) **OCSP Responder Timeout CDF**



(b) **OCSP Average Time to Respond CDF**



(c) **CRL URL Timeout CDF**



(d) **CRL Average Time to Respond CDF**

Figure 7.1: **OCSP, CRL Timeouts and Average Time to Respond**—We show the management of both an intermediate's OCSP responders as well as their CRL distribution points. While the majority of organizations adhere to standards, there are still many poorly managed and broken systems available on the Internet.

poorly managed systems.

The BRs also stipulate that OCSP servers should respond in 10 seconds or less, under normal operating conditions. Figure 7.1b shows a CDF of the average response times for OCSP responders in our dataset. The median response time is 247 milliseconds, but again, with a long tail—the worst OCSP responder takes on average 17.5 seconds to respond to OCSP requests. Still, we note positive progress—a study done in 2012 found that 8.27% of OCSP probes took less than 100ms [20]. We find that 33.4% of our OCSP requests took less than 100ms, and in general, the worst OCSP responders have gotten better over time.

OCSP Correctness: Another important part of maintaining an OCSP responder is ensuring its correctness—any false positives or negatives could negatively impact any application that needs to check the revocation status of certificates. One of the properties spelled out in the BRs is a rule that an OCSP responder should not return a "GOOD" status code for a certificate that was not issued by them. We measure this in the wild, and find that there are 25 OCSP responders that incorrectly respond with "GOOD" for a certificate that was not issued by them.

7.2 Maintenance of the CRL Distribution Point

Although they have been effectively replaced by OCSP responders, Certificate Revocation Lists (CRLs) are still supported by organizations who have customers that rely on them for revocation information. If an organization operates a CRL, there are additional requirements that they must follow in order to be compliant with the BRs, primarily with regards to CRL availability.

CRL Maintenance: Figure 7.1c shows the distribution of timeouts when requesting a CRL in our dataset. We again see a handful of CRL servers that timeout in every request made to them. In addition to being broken, it is also possible that an organization issued certificates with an incorrect CRL URL, such that the timeouts are simply indicative of an organization not updating their CRL service to the right location. This, however, is still a problem, as it makes it impossible for applications to use the provided CRL URL encoded in a certificate to check revocation status without additional information.

CRLs are under the same stipulation as OCSP responders with regards to their response time. Figure 7.1d

Table 7.1: **Correlation Coefficients and P-Values between Different Measurements.**—(GREEN: Strong correlation; RED: Moderate correlation; BLUE: Weak correlation.)

	Errors	Warnings	MDSP Mentions	OCSP Server Mgmt	CRL Server Mgmt.
Errors	-	**0.26** (< 0.01)	**0.26** (0.03)	**0.10** (< 0.01)	0.07 (0.01)
Warnings	**0.26** (< 0.01)	-	**-0.19** (0.06)	**0.19** (< 0.01)	**0.17** (< 0.01)
MDSP Mentions	**0.26** (0.03)	**-0.19** (0.06)	-	**-0.53** (< 0.01)	**-0.17** (0.09)
OCSP Server Mgmt	**0.10** (< 0.01)	**0.19** (< 0.01)	**-0.53** (< 0.01)	-	**0.59** (< 0.01)
CRL Server Mgmt	0.07 (0.01)	**0.17** (< 0.01)	**-0.17** (0.09)	**0.59** (< 0.01)	-

shows the distribution of average response times to a CRL request from the CRL URLs that appear in certificates in our dataset. The median average response time is 252 milliseconds, which is well under the BR's requirement of 10 seconds. However, after this point, the rate of the curve increases rapidly—the latter 50% of CRL URLs are far worse than the former 50%. We find that the latter 50% of CRL URLs belong, on average, to much smaller organizations. The average number of certificates issued by these organizations is 734, compared to an average of 350K certificates issued in the former 50%. This is consistent with our earlier findings about smaller organizations misissuing larger fractions of their certificates.

7.3 The Value of Revocation

Unfortunately, many modern browsers do not check the revocation status of all certificates by default [21]. As a result, some organizations may find it a waste of time and resources to upkeep their revocation infrastructure, despite it being a requirement in the BRs. We argue that the lack of attention and care to revocation serves as an indicator for concerning security attitudes within a PKI organization, and thus is still an adequate measure of organizational management.

7.4 Consistency and Correlation

Our discussion thus far culminates in the question of what relationship, if any, exists between misissuance, management, and an organization's public profile. To quantify these relationships, we use the Spearman's rank correlation test. According to Cohen's guidelines [22], values with absolute correlation coefficients from 0.1 to 0.3 are weakly correlated, from 0.3 to 0.5 are moderately correlated, and greater than 0.5 are considered strongly correlated. Table 7.1 shows the pair-wise correlation coefficients and p-values—of the 10 unique pair-wise values, we find a statistically significant correlation for 90% of the pairs.

First, we investigate the question of consistency within both certificate outputs and management outputs. We find a weak correlation between certificate outputs, indicating that organizations that are likely to make misissuance errors are also likely to cause warnings. There is a strong correlation (0.59) between management outputs, meaning if an organizations mismanages their OCSP responder, they often also mismanage their CRL distribution point. Both of these results are consistent with our hypotheses—we would expect an organization to behave consistently across its many responsibilities.

The PKI community takes misissuance very seriously, primarily because they believe "technical rules are but a proxy for procedure rules" [7]. In other words, adherence to the standards serves as a proxy for how well an organization is managed, and further, both of these features serve as proxies for trustworthiness. We were surprised to find that this anecdote is in fact true—we find there are weak correlations between correctly issuing certificates (ZLint errors and warnings) and organizational management (OCSP, CRL health). One correlation we expected but did not observe was a relationship between ZLint errors, which are strict violations of the requirements, and CRL health. We believe this relationship does not exist because the set of organizations that operate CRLs poorly is small compared to other metrics, and it is possible this skews the correlation result.

The final set of correlations we observe is between misissuance, management, and an organization's public profile, which we gather from MDSP mention data. Because MDSP is the forum where members in the PKI community can report misissuance and other problems with CAs, we expect there to be a strong correlation between both certificate output and MDSP and management indicators and MDSP. Instead, we find only a weak correlation between ZLint errors and MDSP. We noticed that mentions on MDSP has a very long tail—and, as a result, rank ordering in the latter 50% of the dataset is relatively unstable. If we take only the top 50% of the rankings, we observe a correlation of 0.88 with a p-value of $<< 0.01$, indicating a strong positive correlation between ZLint errors and MDSP. We were further surprised by the negative correlation between ZLint warnings and MDSP, which seems to indicate that organizations which cause more warnings are mentioned less. We believe that this is because warnings are not explicit instances of misissuance, and as a result, these certificates are not focused on in online discussions.

We were also surprised by the correlation between MDSP mentions and both OCSP and CRL management,

which is again a negative correlation. In other words, organizations that are worse at management level requirements are mentioned fewer times on MDSP. Our explanation for this is that MDSP is heavily skewed toward large organizations. As we note earlier, smaller organizations are more likely to poorly manage their infrastructure. We find the correlation between total issuance by organization and MDSP mentions is 0.61, with a p-value of 7.4×10^{-5}, indicating a strong tendency to discuss larger players. This result is unfortunate, as it suggests that small players are largely overlooked by the online community, despite the problems that they exhibit.

In general, we find that MDSP is fairly successful in catching certificate misissuance from large organizations, but is missing oversight into smaller organizations. In addition, there is a skew of discussion toward misissuance rather than management, despite management also being a critical part of the baseline requirements. In essence, the community thus far is doing what we believe is the easiest thing—it is far easier and more accessible to determine whether a large intermediate has issued a certificate missing an SAN extension than it is to measure a small organization's OCSP health. We argue that an automated approach to such oversight would better aid the community in catching these problems in the future.

CHAPTER 8

GOING FORWARD

We have seen misissuance serve both as an effective predictor of trustworthiness and of management. As the ecosystem continues to improve, what outstanding problems will we face? In other words, once we remove the worst players from the ecosystem, what remains?

As we note in Chapter 6, there are correlations between an increase in the intermediates/organizations that businesses manage and the number of certificates that each misissues. What contributes to this misissuance? Table 8.1 shows the "largest offending CA" by organization, that is, the intermediate that misissued the largest number of certificates. We find that in 80% of the largest organizations, the majority of error certificates issued can be traced back to exactly one intermediate. For example, the majority of *COMODO's* misissued certificates are generated by the intermediate *COMODO RSA Domain Validation Secure Server*, which accounts for 85.5% of the error certificates. We posit that this is because intermediates within an organization are using entirely disparate codebases or infrastructure in order to issue certificates. This has implications for PKI organizations moving forward—certainly, having disparate issuance processes increases the complexity in issuance. As rules continue to grow and change, maintenance of such processes may become unwieldy, and the surface area for a programming error to manifest will increase. Ultimately, oversight of an organization is harder when there are disparate issuance processes.

Table 8.1: **Intermediate Contribution to Organizational Misissuance**—We show the intermediate that contributes the most to misissuance per organization. In 80% of the organizations that issue more than 10K certificates, the majority of misissued certificates are generated by just one intermediate.

Organization	# Intermediates	Largest Offending Intermediate	% Total Issued	% Total Errors
GoDaddy.com	3	Go Daddy Secure Certification Authority	9%	99.8%
Symantec Corporation	22	Symantec Basic DV SSL CA - G2	60%	94.2%
StartCom Ltd.	17	StartCom Class 1 DV Server CA	83%	76.5%
WoSign CA Limited	39	WoSign CA Free SSL Certificate G2	58%	59.0%
VeriSign	10	VeriSign Class 3 International Server CA - G2	32%	96.1%
GeoTrust Inc.	22	RapidSSL SHA256 CA	52%	49.1%
COMODO CA Limited	29	COMODO RSA Domain Validation Secure Server	22.7%	85.5%
DigiCert Inc	43	DigiCert SHA2 Secure Server CA	52.4%	55.5%
thawte	12	thawte DV SSL CA - G2	30.4%	26.0%
TERENA	9	TERENA SSL CA 3	53.1%	53.2%

Finally, and possibly the hardest problem to catch, is the existence of "one-off" errors—these are errors that are not pervasive within an organization or an intermediate, but appear spuriously due potentially to manual processes or legacy management software. These may appear obviously misissued—for example, a certificate issued for an incorrect TLD or with bad characters in the `dnsName`. What is more dangerous is if these certificates are not obviously misissued, but still incorrect—for example, a test certificate issued for `google.com`, or `example.com`. We see instances of this occurring in our dataset. For instance, the intermediate *thawte SSL CA - G2* issued a test certificate for `*.example.com`, despite not properly validating ownership of the domain before issuance. Moving forward, the community will need to determine how to handle such cases, and identify whether they are serious enough to undermine browser trust.

CHAPTER 9

DISCUSSION

Throughout this work, we discover a number of troubling truths about the CA ecosystem. In this chapter, we draw on the analysis in this work and outline several challenges that we will face as we continue to improve the TLS PKI.

9.1 Security Attitudes in Security Organizations

The mere existence of violations of the RFC and BRs is concerning. While there are instances where the requirements may be ambiguous or difficult to understand, we found most to be well-specified and moderately straightforward to implement. In fact, many if not all such requirements can be easily unit tested. Why is a tool like ZLint not already implemented by each organization? For a task as complex and security-critical as certificate issuance, we would expect at least the creation of the certificate itself to be well tested. Given our understanding that misissuance serves as a proxy for other CA behavior (Chapter 7), we argue that the existence of misissuance speaks to a larger problem regarding security attitudes within PKI organizations.

Our analysis of business owners in Chapter 6 also highlights some outstanding challenges with self-reporting. PKI companies are often bought, sold, and merged, and besides self-reporting through CCADB, there is limited visibility into when this actually occurs. A recent example is the case with WoSign—until WoSign was caught backdating SHA-1 certificates, it was not public knowledge that StartCom was owned by WoSign, as they did not disclose that information through CCADB. The fact that a maliciously motivated security company was able to easily subvert the established guidelines suggests the need for transparency beyond certificates in the ecosystem.

9.2 Consistency in Organizations

The existence of a variety of PKI organizations is good—it reduces the impact of a single point of failure in the TLS ecosystem. However, we argue that consistency within an organization is an important step to operating a well-run and easy to manage system. In Chapter 6, we observe trends that suggest that as CAs grow to encompass more intermediates and organizations, misissuance increases. Further, we find in Chapter 8 that much of this can be attributed to a lack of consistency within an organization. As new requirements are ratified often, we suspect this lack of consistency will become too cumbersome to manage, and that it may pose a risk to the security of the ecosystem. We argue that organizations should work on consistency in managing their intermediates in order to reduce the negative impact they may have on the ecosystem in the future.

9.3 The Value of Measurement

In light of all these outstanding issues, we believe that an automated perspective, like the measurements conducted and presented in this thesis, will increase the visibility of these issues. This is already in play. ZLint now runs as a part of Censys and `crt.sh`, both of which are common tools that the PKI community relies on to inspect and analyze the certificate on the Internet. Though this is a recent addition, the community has already started referencing ZLint on MDSP discussions [23, 24], and some CAs have even floated the idea of including ZLint in the issuance process itself [25]. We further plan to publish management-level information frequently on a web service, so that these classes of problems may also be observed regularly and mitigated.

ZLint, however, is just the first step toward solving this problem. As we note in Chapter 6, misissuance is decreasing over time, and with it, its predictive power will decrease too. We will soon need to focus on measuring more nuanced problems. One suggestion we have is to establish a standardized audit process, where each PKI system is tested against a community devised test-suite, and all results are published in full. There are audit processes currently in place; however, they too are carried out by a variety of third-party companies, and consistency across these third-parties is known to be lacking.

CHAPTER 10

RELATED WORK

Our understanding of the certificate ecosystem has largely been informed by Internet wide scanning of the IPv4 space [18, 26, 27]. As a result, there is a long history of work from both academia and industry that has measured and improved the HTTPS and certificate ecosystem [17, 18, 26, 28–32]. Unfortunately, the ecosystem is becoming increasingly opaque to scanning—VanderSloot et al. showed that scanning can no longer be used as the sole source for certificate measurement, and that we must instead leverage a variety of perspectives to further our understanding. As a result, this study utilizes both Censys [10] and Certificate Transparency Logs [30], which cover 99.4% of the certificates observed by VanderSloot et al. [17].

In addition to large-scale measurements, there has also been focused work on testing the quality of certificate validation in popular libraries [33–35]. These techniques often employ a combination of fuzzing, differential testing, and symbolic execution to determine if there are code paths in validation libraries that are at odds with standards, or worse, introduce bugs that can be exploited by attackers. Most recently, Sivakorn et al. used black-box testing on SSL/TLS libraries in order to check the correctness of the hostname validation process in a variety of client-side libraries [36]. Our work sits on the opposite end of this validation—we investigated the quality of the certificates and the organizations that issue them rather than the quality of the libraries that verify them.

CHAPTER 11

CONCLUSION

In this work, we analyzed the effectiveness of an automated perspective in identifying CA misbehavior. We tracked certificate misissuance, organizational management, and a CA's public profile to provide a holistic view of the problems the ecosystem has solved and the challenges that lie ahead. We additionally built and deployed ZLint, a system for tracking certificate misissuance in the wild. Despite the ecosystem getting better over time, there are still equivalence classes of non-conforming CAs that pose a threat to the security of the PKI. The PKI community is somewhat successful at catching these CAs; however, it suffers from the lack of a systematic approach for observing misbehavior. We provide this work to serve as a guide to researchers and industry alike as we continue to improve the TLS PKI.

REFERENCES

[1] K. Wilson, "Distrusting new WoSign and StartCom certificates," https://blog.mozilla.org/security/2016/10/24/distrusting-new-wosign-and-startcom-certificates/.

[2] D. O'Brien, "Final removal of trust in WoSign and StartCom certificates," https://groups.google.com/a/chromium.org/forum/\#!topic/net-dev/FKXe-76GO8Y.

[3] EFF, "A post mortem on the Iranian diginotar attack," https://www.eff.org/deeplinks/2011/09/post-mortem-iranian-diginotar-attack.

[4] Mozilla, "Revoking trust in two TurkTrust certificates," https://blog.mozilla.org/security/2013/01/03/revoking-trust-in-two-turktrust-certficates/.

[5] Mozilla, "Distrusting new CNNIC certificates," https://blog.mozilla.org/security/2015/04/02/distrusting-new-cnnic-certificates/.

[6] "Baseline requirements documents," https://cabforum.org/baseline-requirements-documents/.

[7] R. Sleevi, "Certificates with invalidly long serial numbers," https://groups.google.com/forum/#!topic/mozilla.dev.security.policy/b33_4CyJbWI.

[8] R. Sleevi, "Certificate transparency in chrome - Change to enforcement date," https://groups.google.com/a/chromium.org/forum/#!msg/ct-policy/sz_3W_xKBNY/6jq2ghJXBAAJ.

[9] "mozilla.dev.security.policy," https://groups.google.com/forum/#!forum/mozilla.dev.security.policy.

[10] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by Internet-wide scanning," in *22nd ACM Conference on Computer and Communications Security*, 2015.

[11] R. Stradling, "crt.sh," https://crt.sh.

[12] D. O'Brien, R. Sleevi, and A. Whalley, "Chromes plan to distrust Symantec certificates," https://security.googleblog.com/2017/09/chromes-plan-to-distrust-symantec.html.

[13] Mozilla, "CA:PROCERT issues," https://wiki.mozilla.org/CA:PROCERT_Issues.

[14] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housely, and P. W., "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," Tech. Rep., 2008.

[15] Mozilla, "Common CA database," http://ccadb.org.

[16] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *22nd USENIX Security Symposium*, 2013.

[17] B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. A. Halderman, "Towards a complete view of the certificate ecosystem," in *16th ACM Internet Measurement Conference*, 2016.

[18] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the https certificate ecosystem," in *13th ACM Internet Measurement Conference*, 2013.

[19] J. Aas, "2017.08.10 Let's Encrypt unicode normalization compliance incident," https://groups.google.com/forum/#!searchin/mozilla.dev.security.policy/nfkc%7Csort:relevance/mozilla.dev.security.policy/nMxaxhYb_iY/AmjCI3_ZBwAJ.

[20] E. Stark, L.-S. Huang, D. Israni, C. Jackson, and D. Boneh, "The case for prefetching and prevalidating tls server certificates," in *19th Network & Distributed Systems Security Symposium*, 2012.

[21] Adam Langley, "Revocation checking and chrome's CRL," https://www.imperialviolet.org/2012/02/05/crlsets.html.

[22] J. Cohen, "Statistical power analysis for the behavioral sciences," Hillsdale, N.J. : L. Erlbaum Associates, 1988.

[23] "Zlint recent error summary," https://misissued.com/zlint.

[24] R. Stradling, "TBSCertificate / Certificate Linting APIs," https://groups.google.com/forum/#!searchin/mozilla.dev.security.policy/ZLint%7Csort:relevance/mozilla.dev.security.policy/sjXswrcsvrE/Nl3OLd4PAAAJ.

[25] G. Markham, "Startcom inclusion request: Next steps," https://groups.google.com/forum/#!searchin/mozilla.dev.security.policy/ZLint%7Csort:relevance/mozilla.dev.security.policy/hNOJJrN6WfE/ocz5eJPAAQAJ.

[26] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "The SSL landscape: A thorough analysis of the X.509 PKI using active and passive measurements," in *11th ACM Internet Measurement Conference*, 2011.

[27] EFF, "SSL Observatory," https://www.eff.org/observatory, 2010.

[28] N. Vratonjic, J. Freudiger, V. Bindschaedler, and J.-P. Hubaux, "The inconvenient truth about web certificates," in *Economics of Security and Privacy III*, 2013.

[29] L. Zhang, D. Choffnes, D. Levin, T. Dumitras, A. Mislove, A. Schulman, and C. Wilson, "Analysis of SSL certificate reissues and revocations in the wake of heartbleed," in *14th ACM Internet Measurement Conference*, 2014.

[30] B. Laurie, A. Langley, and E. Kasper, "Certificate transparency," https://www.certificate-transparency.org/, 2013.

[31] D. Akhawe, B. Amann, M. Vallentin, and R. Sommer, "Here's my cert, so trust me, maybe?: Understanding tls errors on the web," in *22nd International World Wide Web Conference*, 2013.

[32] D. Akhawe and A. P. Felt, "Alice in warningland: A large-scale field study of browser security warning effectiveness," in *22nd USENIX Security Symposium*, 2013.

[33] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, "The most dangerous code in the world: Validating SSL certificates in non-browser software," in *19th ACM Conference on Computer and Communications Security*, 2012.

[34] C. Brubaker, S. Jana, B. Ray, S. Khurshid, and V. Shmatikov, "Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations," in *35th IEEE Symposium on Security and Privacy*, 2014.

[35] S. Y. Chau, O. Chowdhury, E. Hoque, H. Ge, A. Kate, C. Nita-Rotaru, and N. Li, "SymCerts: Practical symbolic execution for exposing noncompliance in X.509 certificate validation implementations," in *38th IEEE Symposium on Security and Privacy*, 2017.

[36] S. Sivakorn, G. Argyros, K. Pei, A. D. Keromytis, and S. Jana, "Hvlearn: Automated black-box analysis of hostname verification in SSL/TLS implementations," in *38th IEEE Symposium on Security and Privacy*, 2017.